

## WEEK 7

**Write a C program to simulate deadlock detection.**

CODE:

```
#include <stdio.h>

int main() {

    int n, m, all[10][10], req[10][10], ava[10], need[10][10];
    int i, j, k, flag[10], prev[10], c, count = 0;

    printf("Enter number of processes and number of resources\n");

    scanf("%d %d", &n, &m);

    printf("Enter total number of required resources %d for each\n", n);
    process\n", n);

    for (i = 0; i < n; i++)

        for (j = 0; j < m; j++)

            scanf("%d", &req[i][j]);

    printf("Enter number of allocated resources %d for each process\n",
n); for (i = 0; i < n; i++)

        for (j = 0; j < m; j++)

            scanf("%d", &all[i][j]);
```

```
printf("Enter number of available resources  
\n"); for (i = 0; i < m; i++)
```

```
    scanf("%d", &ava[i]);
```

```
for (i = 0; i < n; i++)
```

```
    for (j = 0; j < m; j++)
```

```
        need[i][j] = req[i][j] - all[i][j];
```

```
for (i = 0; i < n; i++)
```

```
    flag[i] = 1;
```

```
k = 1;
```

```
while (k) {
```

```
    k = 0;
```

```
    for (i = 0; i < n; i++) {
```

```
        if (flag[i]) {
```

```
            c = 0;
```

```
            for (j = 0; j < m; j++) {
```

```
                if (need[i][j] <= ava[j]) {
```

```
                    c++;
```

```
                }
```

```
            }
```

```
            if (c == m) {
```

```
                for (j = 0; j < m; j++) {
```

```
                    }
```

```

        for (j = 0; j < m; j++) {
            ava[j] += all[i][j];
            all[i][j] = 0;
        }

        flag[i] = 0;
        count++;
    }
}

for (i = 0; i < n; i++) {
    if (flag[i] != prev[i]) {
        k = 1;
        break;
    }
}

for (i = 0; i < n; i++) {
    prev[i] = flag[i];
}

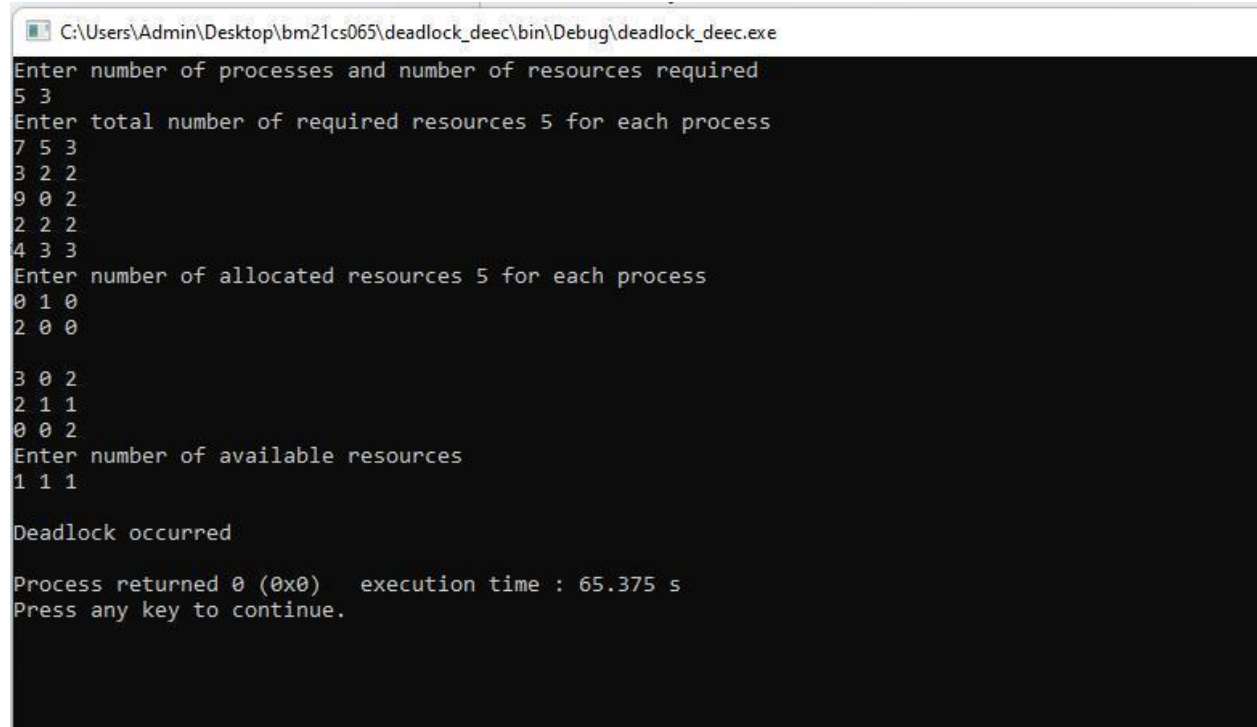
}

if (count == n) {
    printf("\nNo deadlock");
} else {
    printf("\nDeadlock occurred \n");
}

```

```
}  
  
return 0;  
  
}
```

## OUTPUT:



```
C:\Users\Admin\Desktop\bm21cs065\deadlock_deec\bin\Debug\deadlock_deec.exe  
Enter number of processes and number of resources required  
5 3  
Enter total number of required resources 5 for each process  
7 5 3  
3 2 2  
9 0 2  
2 2 2  
4 3 3  
Enter number of allocated resources 5 for each process  
0 1 0  
2 0 0  
  
3 0 2  
2 1 1  
0 0 2  
Enter number of available resources  
1 1 1  
  
Deadlock occurred  
  
Process returned 0 (0x0)   execution time : 65.375 s  
Press any key to continue.
```

